REMARKS

This amendment responds to the Office Action dated October 1, 2002 in which the Examiner rejected claims 1-24 under 35 U.S.C. §103.

Claims 1 and 11 claim a data processing system comprising a plurality of processors and a memory. The plurality of processors execute a series of different types of processing on data to be processed, in a prescribed order. The memory stores the data to be processed in association with state information to represent the processing to be performed next. Processings executed by the plurality of processors are asynchronously executed. The plurality of processors share the memory

Through the structure of the claimed invention having a) processors which execute a series of different types of processing on data and b) a memory which stores state information representing the processing to be next performed, as claimed in claims 1 and 11, the claimed invention provides a data processing system capable of processing data at high speed while allowing the memory capacity to be reduced. The prior art does not show, teach or suggest the invention as claimed in claim 1.

Claims 1-24 were rejected under 35 U.S.C. §103 as being unpatentable over *Orimo et al.* (U.S. Patent No. 5,630,135) in view of *Tanenbaum* (Distributed Operating Systems, 1995).

Orimo et al. appears to disclose the term of "multiple-version programs" means a plurality of programs for performing the same function but having different program structures. The execution results of the programs may be either the same or not the same. (col. 1, lines 18-22) In a distributed processing system having a plurality of processors

connected through a network, at least two first processors execute multiple-version programs which perform the same function, and messages which contain data output as execution results of the programs and attribute information indicating the versions of the executed programs are sent from the first processors to the network. The messages containing the results of processing by the multiple-version programs sent from the first processors are received by a second processor, which selects one message from the received messages based on the attribute information contained in the received messages and executes a program in the second processor by using the data contained in the selected message. (col. 2, lines 2-16) As shown in Fig. 1, multiple-execution system 100 for the multiple-version programs comprises processors 11, 12, 13 and 14 connected to any type of network. Each of the processors 11, 12, 13 and 14 stores an application program in an associated internal memory and executes the application program to conduct various processings. Each application program has a name assigned to correspond to a function and attribute, including version information and is managed thereby. Where the multiple-version programs performing the same function are not present, the attribute thereof is "Null". (col. 3, lines 57-67) FIG. 2 is a diagram of a main part of a format of a message flowing over the network 1. Data transmitted among the processors by the message is stored in a data field 205. A CC field 201 stores a content code indicating the content of the data stored in the data field 205. The processors 11-14 connected to the network determine whether to read in the message flowing over the network 1 or not based on the content code of the CC field 201. A name field 203 stores information for identifying an application program generated by the message. A name of the application

program is used as the information to be stored in the name field 203. An ATR field 204 stores an attribute of the application program generated by the message. (col. 4, lines 1-16)

Thus, *Orimo et al.* merely discloses a plurality of programs for performing the same function but having different program structures. Thus nothing in *Orimo et al.* shows, teaches or suggests processors executing different types of processing as claimed in claims 1 and 11. Rather, *Orimo et al.* teaches away from the claimed invention teaches that the multi-version programs perform the same function (i.e. same processing) but have different program structures.

Additionally, *Orimo et al.* merely discloses a message containing the name of the application program generating the message, an attribute of the application program generated by the message, and a content code indicating the content of the data stored in the data field. Thus nothing in *Orimo et al.* shows, teaches or suggests state information representing the processing to be performed next as claimed in claims 1 and 11. Rather, *Orimo et al.* merely discloses messages storing information about the processing already performed.

Tanenbaum appears to disclose that sharing plays an important role in Mach. No special mechanism is needed to the threads in a process to share objects: they all see the same address space automatically. If one of them has access to a piece of data, they all do. More interesting is the possibility of two or more processes sharing the same memory objects, or just sharing data pages, for that matter. On multiprocessor systems, sharing of objects between two or more processes is frequently even more important. In many cases, a single problem is being solved by a collection of cooperating processes running in

parallel on different CPUs (as opposed to being timeshared on a single CPU). These processes may need access to buffers, tables, or other data structures continuously, in order to do their work. It is essential that the operating system allow this sharing to take place. (page 449)

Thus, *Tanenbaum* merely discloses a shared memory. Nothing in *Tanenbaum* shows, teaches or suggests a) executing different types of processing or b) state information representing the processing to be next performed as claimed in claims 1 and 11. Rather, *Tanenbaum* merely discloses a shared memory.

The combination of *Orimo et al.* and *Tanenbaum* would merely suggest to replace the message discrimination buffer 306 found in each processor as taught by *Orimo et al.* with a shared memory as taught by *Tanenbaum*. Thus nothing in the combination shows, teaches or suggests a) processors executing different types of processing or b) state information representing processing to be performed next as claimed in claims 1 and 11. Therefore, it is respectfully requested that the Examiner withdraws the rejection to claims 1 and 11 under 35 U.S.C. §103.

Claims 2-10 and 12-24 depend from claims 1 and 11 and recite additional features. It is respectfully submitted that claims 2-10 and 12-24 would not have been obvious within the meaning of 35 U.S.C. §103 over *Orimo et al.* and *Tanenbaum* at least for the reasons as set forth above. Therefore, it is respectfully requested that the Examiner withdraws the rejection to claims 2-10 and 12-24 under 35 U.S.C. §103.

Thus it now appears that the application is in condition for reconsideration and allowance. Reconsideration and allowance at an early date are respectfully requested.

Should the Examiner find that the application is not now in condition for allowance, it is respectfully requested that the Examiner enters this amendment for purposes of appeal.

If for any reason Examiner feels that the application is not now in condition for allowance, it is respectfully requested that the Examiner contact, by telephone, the Applicants' undersigned attorney at the indicated telephone number to arrange for an interview to expedite the disposition of this case.

In the event that this paper is not timely filed within the currently set shortened statutory period, Applicants respectfully petitions for an appropriate extension of time. The fees for such extension of time may be charged to our Deposit Account No. 02-4800.

In the event that any additional fees are due with this paper, please charge our Deposit Account No. 02-4800.

Respectfully submitted,

BURNS, DOANE, SWECKER & MATHIS, L.L.P.

By: _____
Ellen Marcie Emas
Registration No. 32,131

P.O. Box 1404
Alexandria, Virginia 22313-1404
(703) 836-6620

Date: December 16, 2002

Mark-up of Claims 1 and 11

1. (Twice Amended) A data processing system comprising:

a plurality of processors for executing a series of different types of processings on data to be processed, in a prescribed order; and

a memory for storing said data to be processed in association with state information to represent the processing [state of said data] to be performed next, wherein

processings executed by said plurality of processors are asynchronously executed and said plurality of processors share said memory.

11. (Twice Amended) A data processing system, comprising:

a plurality of processing means for executing a series of processings of different types on data to be processed, in a prescribed order; and

memory means for storing said data to be processed in association with state information to represent the processing [state of said data] to be next performed, wherein

processings executed by said plurality of processing means are executed asynchronously, and said plurality of processing means share said memory means.